Rule WLM171: Execution velocity was computed on a small sample set

Finding: The service class period being analyzed missed its execution velocity goal. However, the execution velocity was computed on a small sample set. Consequently, the execution velocity might be meaningless.

Impact: This finding is provided for information purposes.

Logic flow: The following rules cause this rule to be invoked:
Rule WLM103: Service class did not achieve execution velocity goal

Discussion: Installations may specify an *execution velocity goal* for a service class period. An execution velocity is a measure of how fast work should run when the work is ready to run, without being delayed waiting for access to a CPU or delayed waiting for access to processor storage. The execution velocity is computed based on samples collected at periodic sampling intervals[1] by the System Resources Manager (SRM). The SRM sampling code interrogates address space control blocks (TCBs, SRBs, OUCBs, and OUXBs) to determine the state of each address space assigned to a service class. Sampling counts associated with the service class are updated based upon the state of the address spaces.

The sampling code records the sampling result into CPU using samples, CPU delay samples, CPU Capping delay samples, and Processor storage delay[2].

Notice that only certain delay categories are included: only delays for processor or for processor storage are included in the "delay" category. These delays are under control of the SRM. Delays not under control of the SRM are not included in CPU or processor storage delays, but are included in an "unknown" delay category. **Unknown delay is not included in the execution velocity computation**. The "unknown" delay means that the SRM was unable to identify the cause of delay. In practice, this means that the delay was something over which the SRM had no control (e.g., I/O operations, ENQ delay, etc.).

[1]With MVS/ESA SP5.1, the sampling interval is 250 milliseconds. The state of each TCB or SRB associated with an address space is sampled every 250 milliseconds, beginning from address space initiation.

[2]Processor storage delay samples means that an address space is ready to execute, but is delayed waiting for processor storage. Eight separate processor storage delays are recorded (swap-in delay, MPL delay, and six categories of page-in delay from auxiliary storage)

The Workload Manager computes the execution velocity of a service class by applying the following algorithm:

$$\frac{\textit{using samples}}{\textit{using samples} \% \textit{delay samples}} (100$$

where:

using samples include:

- The number of samples of work using the processor (CPU Using).
- The number of calculated samples of work using non-paging DASD I/O resources (DASD connect state or DASD disconnect state). I/O using samples are included only if the installation has elected to include WLM-managed I/O.

delay samples include:

- The number of samples of work delayed for the processor (Denied CPU Delay or CPU Capping delay).
- The number of samples of work delayed for processor storage. Delay for processor storage includes:
  - Paging delay
  - Swap-in delay
  - Swapped out for multiprogramming (MPL) reasons
  - Server address space creation delay
  - Initiation delays for batch jobs in WLM-managed job classes
- The number of calculated samples of work delayed for non-paging DASD I/O resources (DASD IOS queue delay, DASD subchannel pending delay, or DASD control unit queue delay). I/O delay samples are included only if the installation has elected to include WLM-managed I/O.

The result from the algorithm is multiplied by 100, to yield an execution velocity ranging from 0 (when the address space did not use the CPU) to

100 (when the address space was not delayed for any reason controlled by the SRM).

It is important to keep in mind that execution velocity applies **only to times when an address space is using a CPU or ready to use a CPU**. It does not include times when an address space is idle, waiting for I/O, enqueued for a resource, etc. The SRM takes samples every 250 milliseconds, or 4 times per second. If the address spaces in a service class are idle or waiting for some unknown reason for most of the time, the SRM might not be able to collect sufficient samples to compute a valid execution velocity.

The following example illustrates the problem:

- Suppose that the address spaces in a service class are idle or waiting for unknown reasons for 95% of the time. This behavior is common with some Started Tasks (such as VTAM, RMF, etc.) During only 5% of the time, would the SRM find the address spaces in one of the states that contribute to execution velocity (Using CPU, CPU delay, processor storage delay).

- During the 10-second policy adjustment interval, the Workload Manager would have only 2 samples for the previous interval (4 samples per second * 10 seconds * 0.05 = 2).

- The Workload Manager normally keeps about 20 minutes history information. Over an entire 20 minute interval, the SRM would collect only 240 samples (20 minutes * 60 seconds per minute * 4 samples per second * 0.05 = 240).

- While 240 samples might be a sufficiently large number to yield a valid result, recall that this value is an accumulation over 20 minutes and Workload Manager decisions would necessarily assume that the 20 minutes' data represent the CPU demand and delays of the address spaces in the service class.

More insidious is the fact that, beginning with MVS/ESA SP3.1, the MVS Dispatcher algorithms were redesigned to implement a "reduced preemption" technique of dispatching[3]. With reduced preemption, a newly-ready task at a high dispatching priority might not immediately interrupt a task at a lower dispatching priority. Rather, dispatching operates on a "time-sliced" basis, and the interrupt might be delayed for a short time[4] before the Dispatcher proceeds with the interrupt. This algorithm was

[3]Lambourne (see reference) provides an excellent discussion of the full versus reduced/partial preemption algorithms.

[4]The delay is dynamically adjusted by the SRM, but typically varies between 1 and 5 milliseconds.

implemented to achieve greater benefit from the very high speed processor cache registers delivered with modern IBM processors.

- Reduced preemption has a significant effect on execution velocity. The tasks that are mostly idle (for example, Started Tasks) tend to use the processor is short bursts (that is, they are idle for a long percent of their elapsed time but want to use the processor when they become ready to execute). The tasks typically have a low mean time to wait when they are ready (that is, they use the CPU for a short time, then relinquish the CPU for I/O activity).

- If a Started Task uses only 100 microseconds of CPU time per dispatch and the average time between becoming ready and being dispatched is 2 milliseconds (because of reduced preemption), over 95% of the time the Started Task ready time would be waiting for CPU (CPU Delay). This time would translate into an **achievable** execution velocity of less than 5 (100÷(2000+100)=4.76), regardless of the execution velocity goal specified for the service class!

  The Workload Manager could not achieve a high execution velocity goal for this type of task, even though the Started Task had been assigned a high dispatch priority. This is an effect of the basic Dispatcher algorithms rather than the Workload Manager algorithms.

When CPExpert determines that a service class with an execution velocity goal has missed its performance goal, CPExpert reviews the number of samples on which the execution velocity is based. CPExpert produces Rule WLM171 if the number of samples is small.

Once CPExpert has determined that an unacceptably small number of samples exist, no further analysis is done. It makes no sense to analyze delays to the service class based on a low number of samples, inasmuch as the conclusions from the samples would be invalid.

The following example illustrates the output from Rule WLM171:

```
RULE WLM171:  EXECUTION VELOCITY WAS COMPUTED ON A SMALL SAMPLE SET

   The delay information presented above is based on the CPU Using and
   Execution Delay samples of the ASCH Service Class (execution velocity
   is based on these samples).  These percentages show the distribution of
   time when an address space in the service class was executing (using the
   CPU, waiting to use the CPU, or waiting for processor storage).  For a
   significant percent of their overall active time, address spaces in the
   ASCH Service Class were either IDLE or were waiting on some event not
   included in the execution velocity calculations.  The below information
   shows the percent of total active time in which address spaces in this
   service class were executing, were delayed for UNKNOWN reasons, or were
   idle.  Please refer to Rule WLM171 in the WLM Component User Manual for
   discussion of the implications of this finding.

                            AVERAGE      PCT         PCT       PCT    EXECUTION
   MEASUREMENT INTERVAL       MPL      EXECUTING   UNKNOWN    IDLE     SAMPLES
   14:30-14:45,01MAR1994      1.0        0.0         0.0     100.0       3
   14:45-15:00,01MAR1994      1.0        0.1         0.0      99.9       5
```

Suggestion: CPExpert suggests that you consider the following alternatives:

- If the service class reported by Rule WLM171 consists of Started Tasks, you should assess the important of the Started Tasks.

  - If the Started Tasks are important from a system view (e.g., VTAM), you should consider allowing the Started Tasks to default to the SYSSTC service class. The SYSSTC service class has a high dispatching priority. Address spaces in SYSSTC will not be subject to the Workload Manager's execution velocity algorithms[5].

  - If the Started Tasks are not important from a system view, you should consider removing them from a service class with relatively high execution velocity goals (since the Workload Manager is unable to achieve the goals). You may wish to assign them to a service class with (1) relatively low execution velocity goals or (2) discretionary goals.

- If you are comfortable with the current placement of address spaces in the service class reported by Rule WLM171, you should consider excluding the service class from analysis by CPExpert (using the EXCLUDE guidance parameters described in Section 2 (Chapter 1.1.8) of this document). You likely would become annoyed by CPExpert continually reporting that the service class missed its performance goal when you contemplate no action.

[5]This alternative does not reduce the effect of the reduced preemption on address spaces in the service class. The alternative simply removes them from the Workload Manager's control.

- Alternatively, you can provide different guidance to CPExpert's analysis by altering the EXECSAMP guidance variable in USOURCE(WLMGUIDE).

Reference: "MVS/ESA Full vs. Reduced/Partial Preemption", Lambourne, Steve, *1994 Proceedings of the Computer Measurement Group*, page 1347.

IBM TalkLink MVSWLM CFORUM, Appended at 19:23:56 on 10/24/96 by DISKER at KGNVMC (John Arwe, SRM/WLM Development Team).

